

# **Отчёт по лабораторной работе № 6**

**Дисциплина: Архитектура компьютера**

Румянцев Артём Олегович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Символьные и численные данные в NASM . . . . .	9
4.2	Выполнение арифметических операций в NASM . . . . .	14
4.2.1	Ответы на вопросы по программе . . . . .	18
4.3	Выполнение заданий для самостоятельной работы . . . . .	19
<b>5</b>	<b>Выводы</b>	<b>22</b>
<b>6</b>	<b>Список литературы</b>	<b>23</b>

## Список иллюстраций

4.1	Создание директории . . . . .	9
4.2	Создание файла . . . . .	9
4.3	Создание копии файла . . . . .	10
4.4	Редактирование файла . . . . .	10
4.5	Запуск исполняемого файла . . . . .	10
4.6	Редактирование файла . . . . .	11
4.7	Запуск исполняемого файла . . . . .	11
4.8	Создание файла . . . . .	12
4.9	Редактирование файла . . . . .	12
4.10	Запуск исполняемого файла . . . . .	12
4.11	Редактирование файла . . . . .	13
4.12	Запуск исполняемого файла . . . . .	13
4.13	Редактирование файла . . . . .	13
4.14	Запуск исполняемого файла . . . . .	14
4.15	Создание файла . . . . .	14
4.16	Редактирование файла . . . . .	15
4.17	Запуск исполняемого файла . . . . .	15
4.18	Изменение программы . . . . .	16
4.19	Запуск исполняемого файла . . . . .	16
4.20	Создание файла . . . . .	16
4.21	Редактирование файла . . . . .	17
4.22	Запуск исполняемого файла . . . . .	17
4.23	Создание файла . . . . .	19
4.24	Написание программы . . . . .	19
4.25	Запуск исполняемого файла . . . . .	20
4.26	Запуск исполняемого файла . . . . .	20

## Список таблиц

# 1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

### 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`. - Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`. - Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного

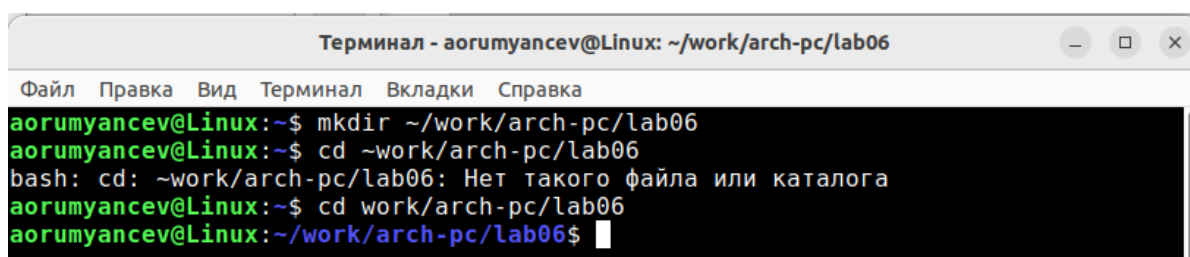
результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно



## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

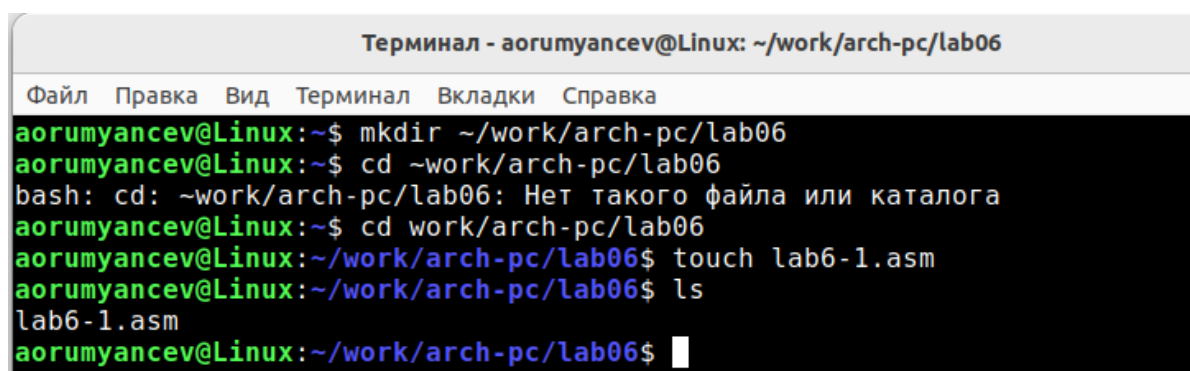
С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. 01). Перехожу в созданный каталог с помощью утилиты `cd`.



```
Терминал - aorummyancev@Linux: ~/work/arch-pc/lab06
Файл  Правка  Вид  Терминал  Вкладки  Справка
aorummyancev@Linux:~$ mkdir ~/work/arch-pc/lab06
aorummyancev@Linux:~$ cd ~/work/arch-pc/lab06
bash: cd: ~/work/arch-pc/lab06: Нет такого файла или каталога
aorummyancev@Linux:~$ cd work/arch-pc/lab06
aorummyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.1: Создание директории

С помощью утилиты `touch` создаю файл `lab6-1.asm` (рис. 02).



```
Терминал - aorummyancev@Linux: ~/work/arch-pc/lab06
Файл  Правка  Вид  Терминал  Вкладки  Справка
aorummyancev@Linux:~$ mkdir ~/work/arch-pc/lab06
aorummyancev@Linux:~$ cd ~/work/arch-pc/lab06
bash: cd: ~/work/arch-pc/lab06: Нет такого файла или каталога
aorummyancev@Linux:~$ cd work/arch-pc/lab06
aorummyancev@Linux:~/work/arch-pc/lab06$ touch lab6-1.asm
aorummyancev@Linux:~/work/arch-pc/lab06$ ls
lab6-1.asm
aorummyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.2: Создание файла

Копирую в текущий каталог файл in\_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис.03).

```
aorumyancev@Linux:~/work/arch-pc/lab06$ cp ~/Загрузки/in_out.asm in_out.asm
aorumyancev@Linux:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm
aorumyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.3: Создание копии файла

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. 04).



Рис. 4.4: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 05). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

```
aorumyancev@Linux:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aorumyancev@Linux:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
aorumyancev@Linux:~/work/arch-pc/lab06$ ./lab6-1
j
aorumyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.5: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 06).

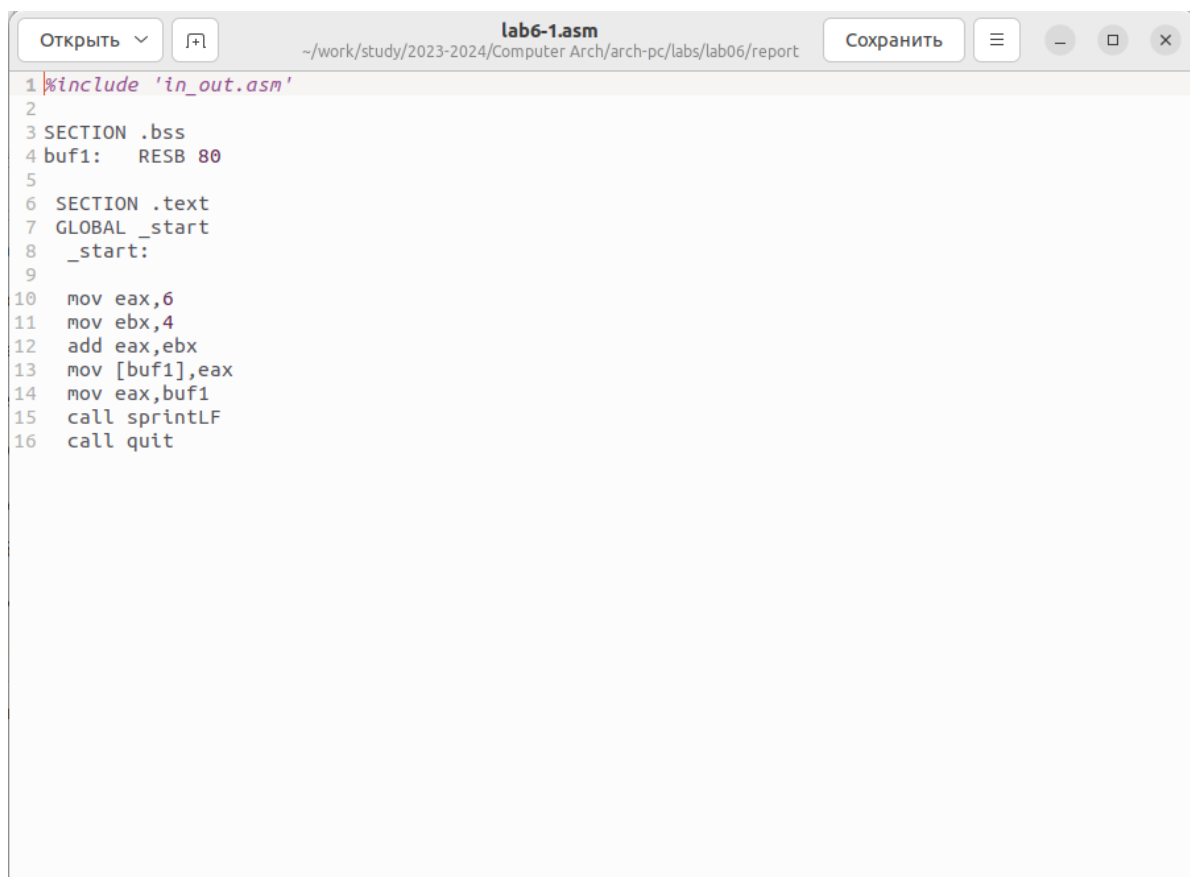


Рис. 4.6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 07). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

```
aorумыancev@Linux:~/work/arch-pc/lab06$ open lab6-1.asm
aorумыancev@Linux:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aorумыancev@Linux:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
aorумыancev@Linux:~/work/arch-pc/lab06$ ./lab6-1

aorумыancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.7: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 08).

```

aorumyancev@Linux:~/work/arch-pc/lab06$ touch lab6-2.asm
aorumyancev@Linux:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm

```

Рис. 4.8: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 09).



```

1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 call iprintLF
11
12 call quit

```

Рис. 4.9: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. 10). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

```

aorumyancev@Linux:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aorumyancev@Linux:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aorumyancev@Linux:~/work/arch-pc/lab06$ ./lab6-2
106
aorumyancev@Linux:~/work/arch-pc/lab06$

```

Рис. 4.10: Запуск исполняемого файла

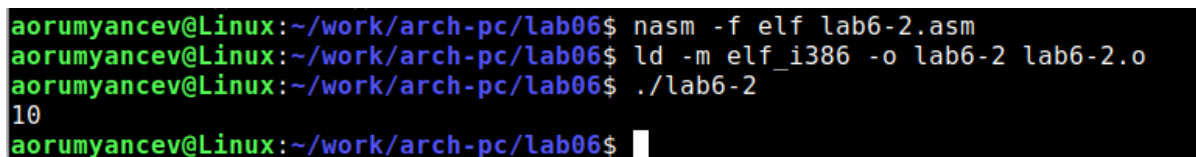
Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4 (рис. 11).



```
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprintLF
11
12 call quit
```

Рис. 4.11: Редактирование файла

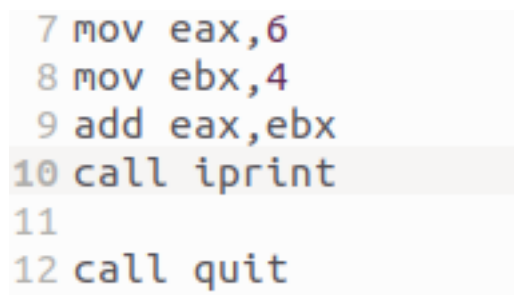
Создаю и запускаю новый исполняемый файл (рис. 12). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.



```
aorumyancev@Linux:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aorumyancev@Linux:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aorumyancev@Linux:~/work/arch-pc/lab06$ ./lab6-2
10
aorumyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.12: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 13).



```
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprint
11
12 call quit
```

Рис. 4.13: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 14). Вывод изменился, в отличии от предыдущей команды iprintLF, iprint выводит значение, но не переносит ввод на следующей строке, а остаётся на этой же. iprintLF добавляет

символ переноса строки, а `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`.

```
aorumyancev@Linux:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aorumyancev@Linux:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aorumyancev@Linux:~/work/arch-pc/lab06$ ./lab6-2
10aorumyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.14: Запуск исполняемого файла

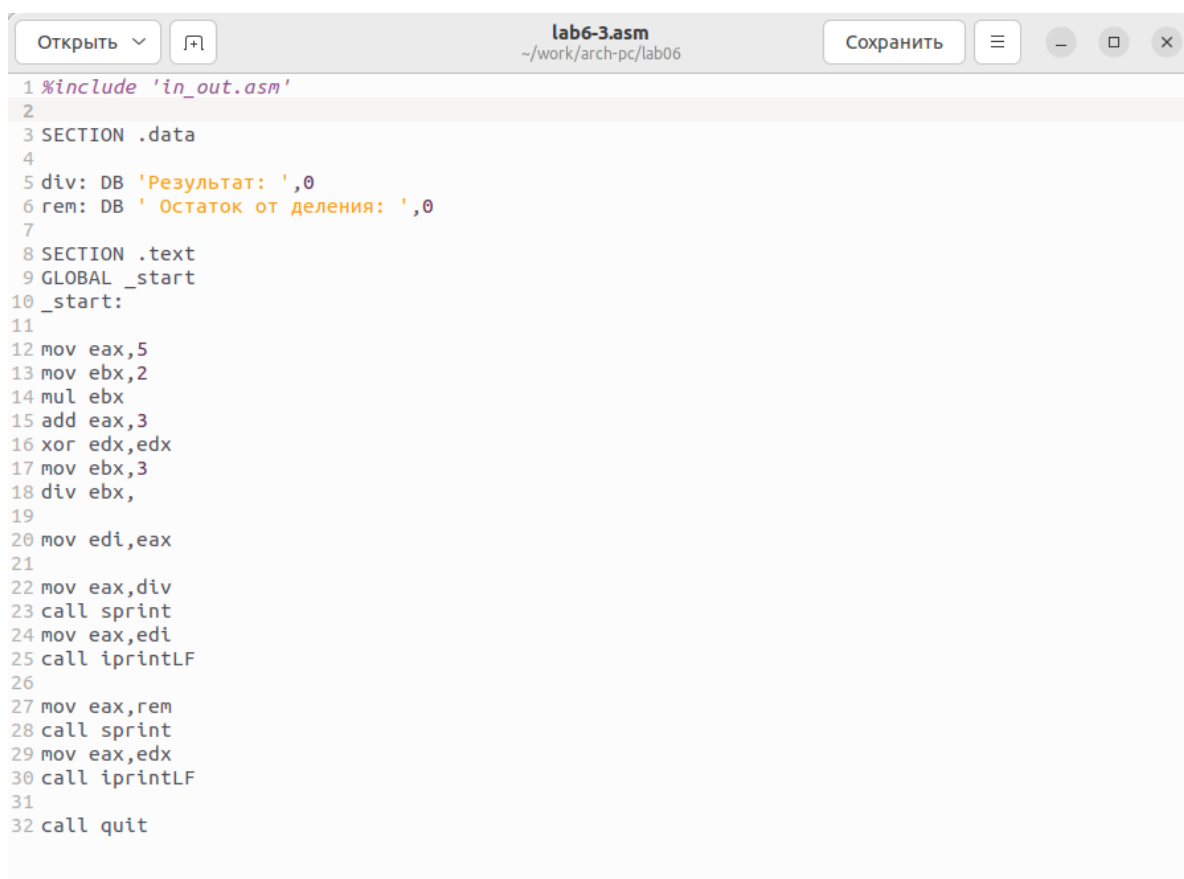
## 4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch` (рис. 15).

```
10aorumyancev@Linux:~/work/arch-pc/lab06$ touch lab6-3.asm
aorumyancev@Linux:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o
lab6-1      lab6-1.o    lab6-2.asm  lab6-3.asm
aorumyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.15: Создание файла

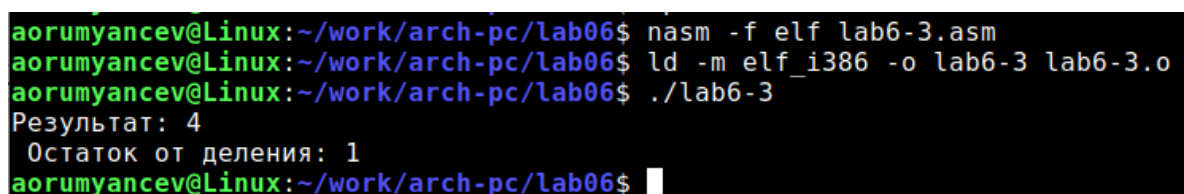
Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 * 2 + 3)/3$  (рис. 16).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4
5 div: DB 'Результат: ',0
6 rem: DB 'Остаток от деления: ',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 mov eax,5
13 mov ebx,2
14 mul ebx
15 add eax,3
16 xor edx,edx
17 mov ebx,3
18 div ebx
19
20 mov edi,eax
21
22 mov eax,div
23 call sprint
24 mov eax,edi
25 call iprintLF
26
27 mov eax,rem
28 call sprint
29 mov eax,edx
30 call iprintLF
31
32 call quit
```

Рис. 4.16: Редактирование файла

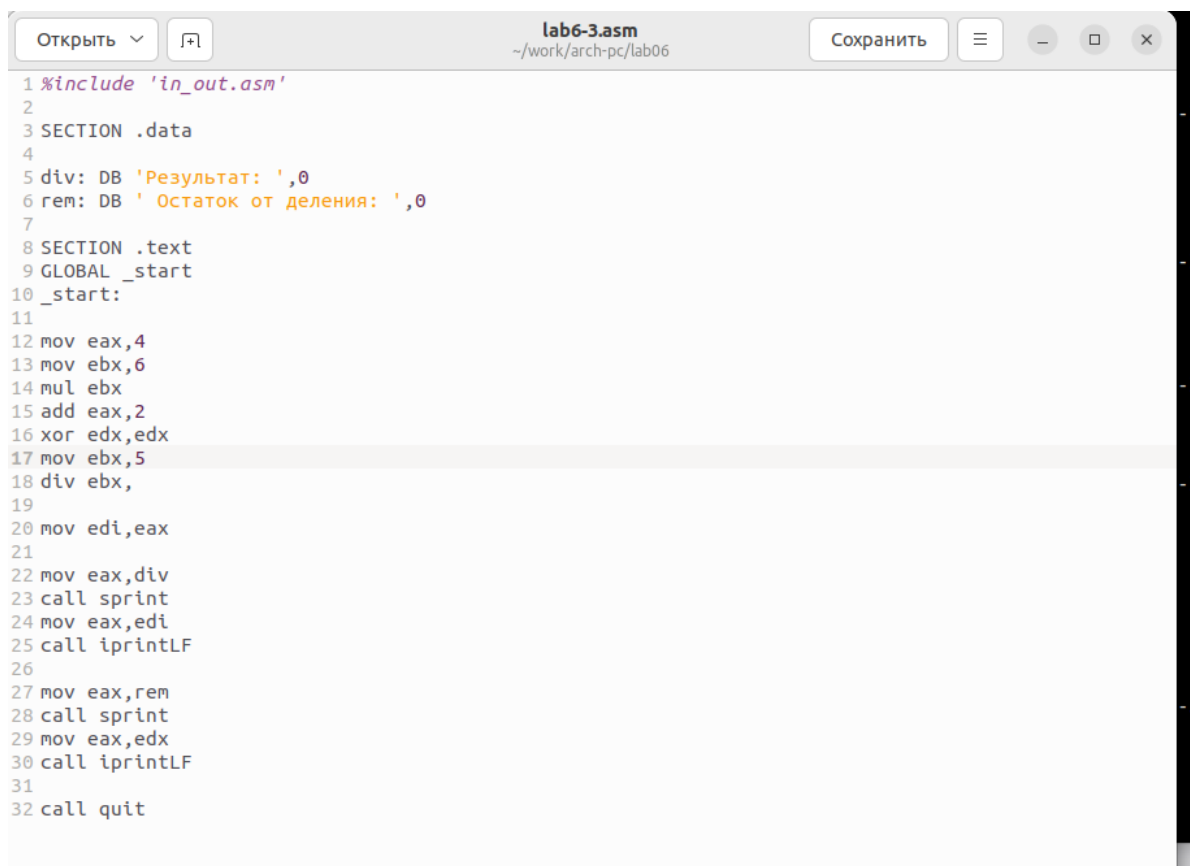
Создаю исполняемый файл и запускаю его (рис. 17).



```
aorummyancev@Linux:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aorummyancev@Linux:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
aorummyancev@Linux:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
aorummyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.17: Запуск исполняемого файла

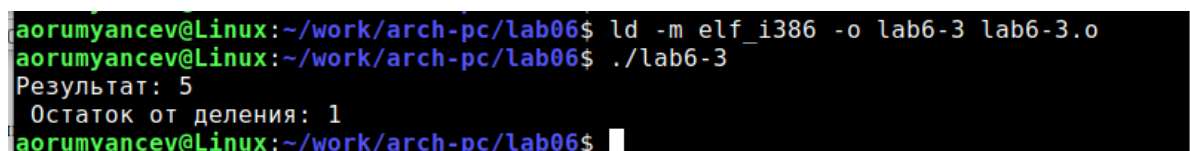
Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4 * 6 + 2)/5$  (рис. 18).



```
1 %include 'in_out.asm'
2
3 SECTION .data
4
5 div: DB 'Результат: ',0
6 rem: DB ' Остаток от деления: ',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 mov eax,4
13 mov ebx,6
14 mul ebx
15 add eax,2
16 xor edx,edx
17 mov ebx,5
18 div ebx,
19
20 mov edi,eax
21
22 mov eax,div
23 call sprint
24 mov eax,edi
25 call iprintLF
26
27 mov eax,rem
28 call sprint
29 mov eax,edx
30 call iprintLF
31
32 call quit
```

Рис. 4.18: Изменение программы

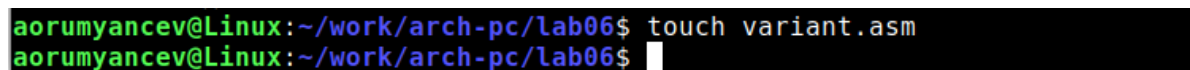
Создаю и запускаю новый исполняемый файл (рис. 19). Я посчитал для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.



```
aorumyancev@Linux:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
aorumyancev@Linux:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
aorumyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.19: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 20).



```
aorumyancev@Linux:~/work/arch-pc/lab06$ touch variant.asm
aorumyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.20: Создание файла

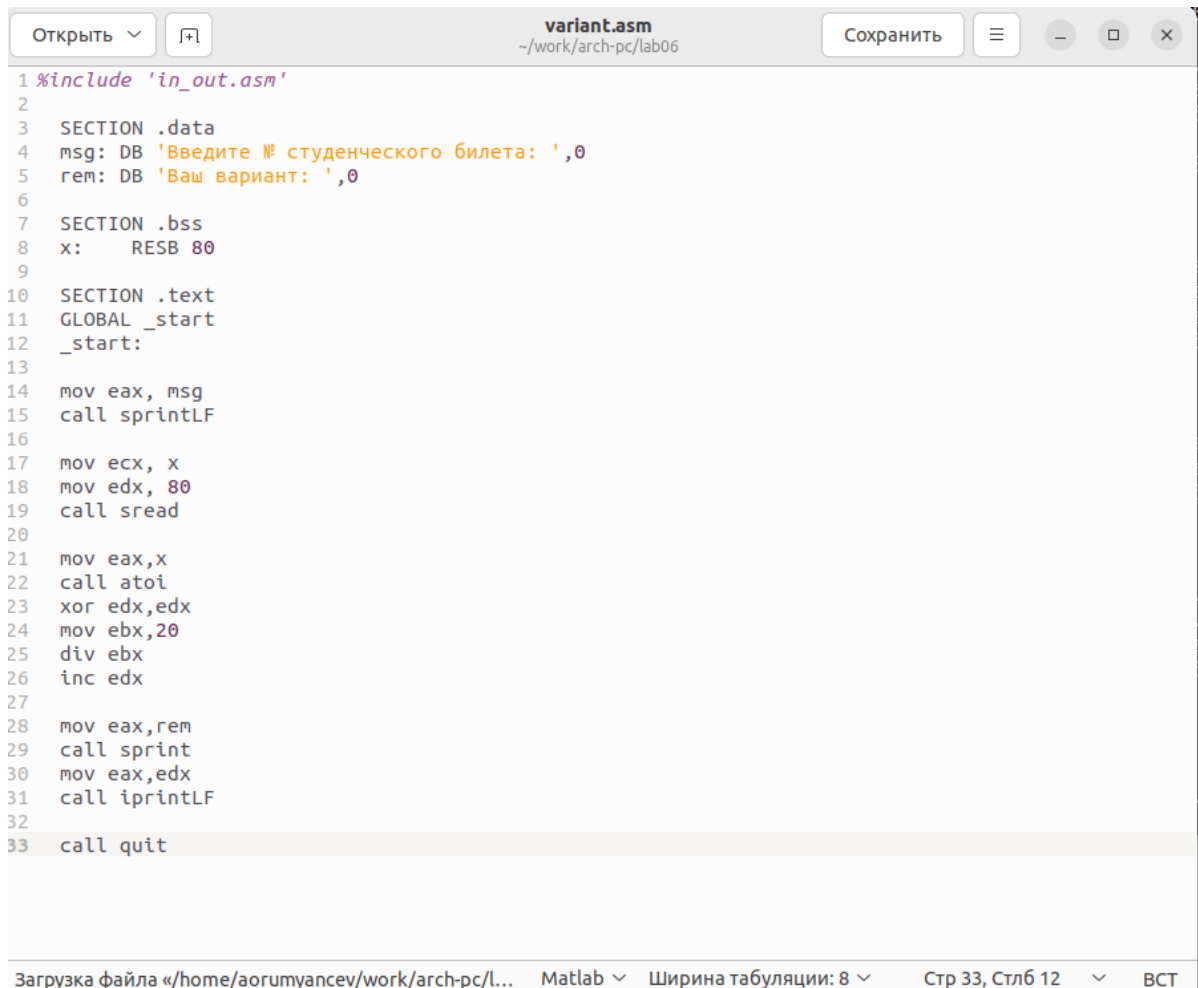


Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 21).

```
aorumyancev@Linux:~/work/arch-pc/lab06$ nasm -f elf variant.asm
aorumyancev@Linux:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
aorumyancev@Linux:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132231426
Ваш вариант: 7
aorumyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.21: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 22). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 8.



```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg: DB 'Введите № студенческого билета: ',0
5 rem: DB 'Ваш вариант: ',0
6
7 SECTION .bss
8 x:   RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13
14 mov eax, msg
15 call sprintf
16
17 mov ecx, x
18 mov edx, 80
19 call sread
20
21 mov eax, x
22 call atoi
23 xor edx, edx
24 mov ebx, 20
25 div ebx
26 inc edx
27
28 mov eax, rem
29 call sprintf
30 mov eax, edx
31 call iprintLF
32
33 call quit
```

Рис. 4.22: Запуск исполняемого файла

## 4.2.1 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov eax,rem  
call sprint
```

2. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки  
call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
4. За вычисления варианта отвечают строки:

```
xor edx,edx ; обнуление edx для корректной работы div  
mov ebx,20 ; ebx = 20  
div ebx ; eax = eax/20, edx - остаток от деления  
inc edx ; edx = edx + 1
```

5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx
6. Инструкция inc edx увеличивает значение регистра edx на 1
7. За вывод на экран результатов вычислений отвечают строки:

```
mov eax,edx  
call iprintLF
```

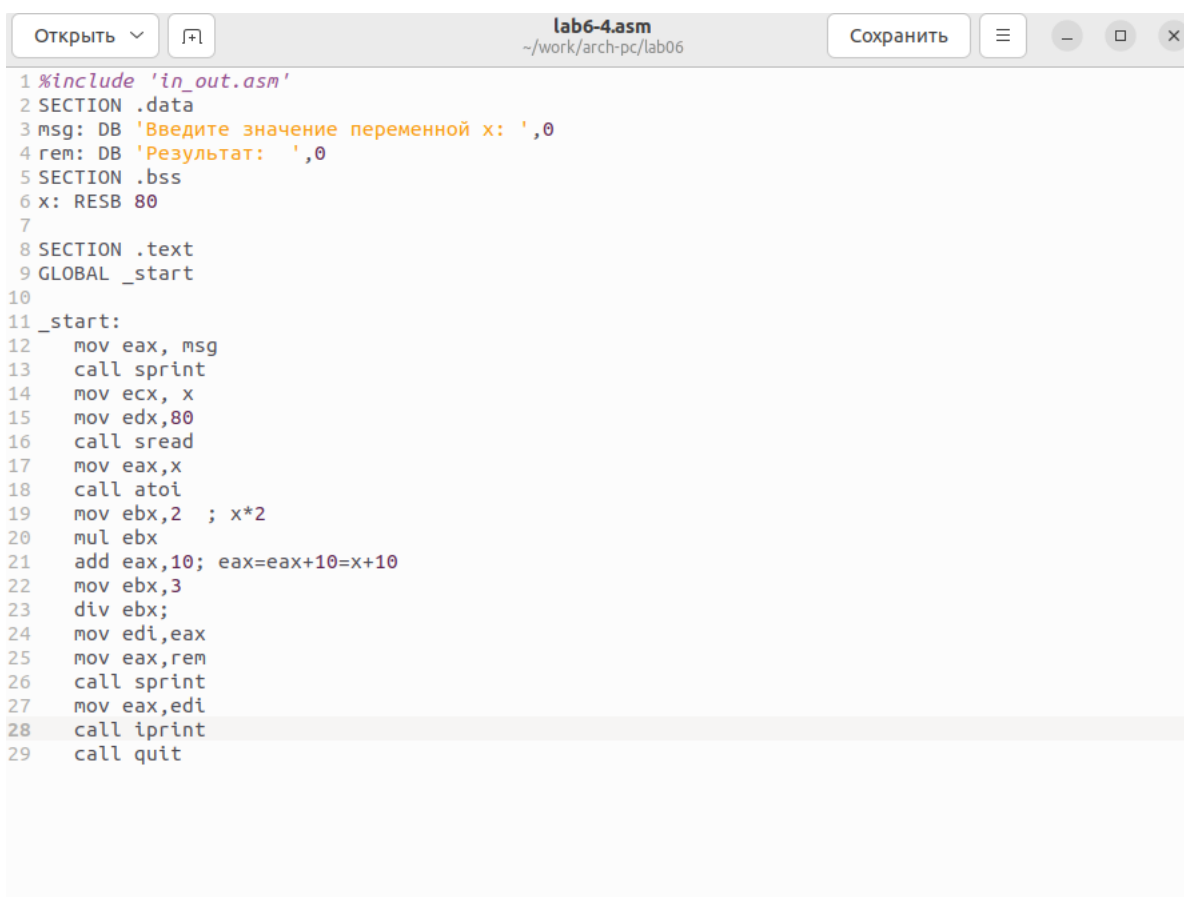
## 4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch (рис. 23).

```
aorumyancev@Linux:~/work/arch-pc/lab06$ touch lab6-4.asm
aorumyancev@Linux:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.o    lab6-2.o    lab6-3.o    variant.asm
lab6-1      lab6-2     lab6-3     lab6-4.asm  variant.o
lab6-1.asm  lab6-2.asm lab6-3.asm  variant
aorumyancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения  $(10 + 2x) / 3$  (рис. 24). Это выражение было под вариантом 1.



```
lab6-4.asm
~/work/arch-pc/lab06
Сохранить

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5 SECTION .bss
6 x: RESB 80
7
8 SECTION .text
9 GLOBAL _start
10
11 _start:
12     mov eax, msg
13     call sprint
14     mov ecx, x
15     mov edx, 80
16     call sread
17     mov eax, x
18     call atoi
19     mov ebx, 2 ; x*2
20     mul ebx
21     add eax, 10; eax=eax+10=x+10
22     mov ebx, 3
23     div ebx;
24     mov edi, eax
25     mov eax, rem
26     call sprint
27     mov eax, edi
28     call iprint
29     call quit
```

Рис. 4.24: Написание программы

Создаю и запускаю исполняемый файл (рис. 25). При вводе значения 10, вывод

- 10.

```
aorумыancev@Linux:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
aorумыancev@Linux:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
aorумыancev@Linux:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 10
Результат: 10aorумыancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.25: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 26). Программа отработала верно.

```
aorумыancev@Linux:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 1
Результат: 4aorумыancev@Linux:~/work/arch-pc/lab06$
```

Рис. 4.26: Запуск исполняемого файла

**Листинг 4.1. Программа для вычисления значения выражения  $(10 + 2x) / 3$ .**

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
```

```
call atoi
mov ebx,2 ; x*2
mul ebx
add eax,10; eax=eax+10=x+10
mov ebx,3
div ebx;
mov edi,eax
mov eax,rem
call sprint
mov eax,edi
call iprint
call quit
```

## 5 Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера NASM.

## 6 Список литературы

1. Лабораторная работа №6
2. Таблица ASCII